



# Naval Ship Database

*Database Design, Implementation, and Schema*

Christopher Woo  
Co-op Student  
DRDC CORA

DRDC CORA TN 2013–157  
September 2013

**Defence R&D Canada**  
**Centre for Operational Research and Analysis**

Maritime Operational Research Team  
Director of Maritime Strategy



National  
Defence

Défense  
nationale

**Canada**



# **Naval Ship Database**

*Database Design, Implementation, and Schema*

*Christopher Woo  
Coop Student  
DRDC CORA*

*Supervisor:  
Ramzi Mirshak and Paul Massel  
Maritime Operational Research Team*

## **Defence R&D Canada – CORA**

Technical Note  
DRDC CORA TN 2013-157  
September 2013

Principal Author

*Original signed by Christopher Woo*

---

Christopher Woo

Defence Scientist

Approved by

*Original signed by Dr. R.E. Mitchell*

---

Dr. R.E. Mitchell

Head Maritime and Air Systems Operational Research

Approved for release by

*Original signed by Paul Comeau*

---

Paul Comeau

DRDC CORA Chief Scientist

The information contained herein has been derived and determined through best practice and adherence to the highest levels of ethical, scientific, and engineering investigative principles. The reported results, their interpretation, and any opinions expressed therein, remain those of the author and do not represent or otherwise reflect any official opinion or position of DND or the Government of Canada.

Defence R&D Canada – Centre for Operational Research and Analysis (CORA)

© Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2013

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2013

## Abstract

---

This Technical Note documents a database that was designed and implemented to manage a collection of historical data housed by the Maritime Operational Research Team (MORT), which is part of Defence Research and Development Canada's Center for Operational Research and Analysis (DRDC CORA). The goals of the project included: improving integrity; allowing scalability; simplifying queries across existing datasets; and providing schema flexibility for new incoming data. The solution allows database users to store and analyze data collected by navy ships in the Royal Canadian Navy (RCN). The data stored in the database includes but is not limited to ship identification, ship log records, daily scheduled activities, distance made good, designated homeport, geolocation, readiness status, timezone reference changes, and observed weather. Multiple database storage solutions were compared and SQL Server was selected to be the one most suitable for the Naval Ship Database project. Future projects should consider SQL Server as the primary data storage system.

## Résumé

---

La présente note technique décrit une base de données été conçue et mise en œuvre pour gérer un ensemble de données historiques hébergées par l'équipe de recherche opérationnelle des Forces maritimes, qui fait partie du Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC). Le projet vise à améliorer l'intégrité et l'extensibilité, à simplifier les requêtes dans des ensembles de données existants et à fournir des schémas adaptables qui peuvent accueillir de nouvelles données. La solution mise en place permet aux utilisateurs de stocker et d'analyser des données recueillies par des navires de la Marine royale canadienne (MRC). Sont stockées dans la base de données : données d'identification des navires, entrées de journal de bord, activités quotidiennes prévues, distances couvertes, ports d'attache désignés, géolocalisation, statuts opérationnels, changements de fuseau horaire de référence, données météorologiques observées, etc. Nous avons comparé de multiples solutions de bases de données et déterminé que SQL Server était la solution la mieux adaptée au projet de base de données de navires de la Marine. Les projets à venir devraient envisager l'utilisation de SQL Server comme système principal de stockage de données.

This page intentionally left blank.

## Executive summary

---

### Naval Ship Database: Database Design, Implementation, and Schema

**Christopher Woo; DRDC CORA TN 2013-157; Defence R&D Canada – CORA; September 2013.**

**Introduction or background:** Over the last few years, the Maritime Operational Research Team (MORT) of Defence Research and Development Canada Centre for Operational Research and Analysis (DRDC CORA) has been collecting various datasets that document historical ship activities. Combining that information into a database has been recognized as a beneficial step forward in its data management, and will simplify efforts for ongoing efforts within MORT as it undertakes the Frigate Ship's Boat Usage Study for Director Naval Requirements (See Reference [1]). Following past efforts, the Naval Ship Database was designed and implemented, making improvements on design flaws that existed in the previously a designed solution [2].

**Results:** The new dataset combines information on ship positions, activities, and log entries. SQL Server was selected for implementation due to its availability, robustness, powerful querying ability, flexibility for project extensions, and compatibility with other systems within DRDC and the Department of National Defence (DND).

**Significance:** The Naval Ship Database will prove to be a step forward for MORT in data collection and storage efforts. Data can now be obtained in electronic form and quickly queried with the flexible and powerful SQL standard. The project can also open new methodologies in collecting and organizing data by exploring machine learning applications.

**Future plans:** The database will be expanded as more features are analyzed and more data are collected.

# Sommaire

## Naval Ship Database: Database Design, Implementation, and Schema

**Christopher Woo ; DRDC CORA TN 2013-157 ; R & D pour la défense Canada – CARO; septembre 2013.**

**Introduction ou contexte :** Au cours des dernières années, l'équipe de recherche opérationnelle des Forces maritimes (MORT), du Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC), a recueilli divers ensembles de données relatives aux activités historiques des navires. Il a été reconnu qu'il serait utile de rassembler ces données dans une base de données pour aider à les gérer et simplifier les efforts continus au sein de la MORT d'étude sur l'utilisation des frégates à l'intention du directeur – Besoins navals (voir référence [1]). À la suite d'efforts précédents, la Base de données des navires de la Marine a été conçue et mise en œuvre et a corrigé des défauts de conception qui existaient dans la solution précédente [2].

**Résultats :** Le nouvel ensemble de données rassemble des données de positions, d'activités et d'entrées de journal de bord de navires. Nous avons choisi SQL Server pour la mise en œuvre de la base de données, compte tenu de sa disponibilité, de sa fiabilité, de ses puissantes capacités d'interrogation, de son adaptabilité en vue d'élargir le projet et de sa compatibilité avec d'autres systèmes de RDDC et du Ministère de la Défense nationale (MDN).

**Importance :** La base de données sur les navires de la marine s'avérera un pas dans la bonne direction pour la collecte et le stockage de données de l'équipe MORT. Les données sont maintenant disponibles en ligne et peuvent être interrogées rapidement au moyen de la souple et puissante interface SQL normalisée. Le projet permet aussi d'explorer de nouvelles méthodes de collecte et d'organisation des données grâce aux applications d'apprentissage automatique.

**Perspectives :** La base de données s'élargira au fur et à mesure de l'ajout de nouvelles fonctions et de la collecte d'autres données.



# Table of contents

---

Abstract .....	i
Résumé .....	i
Executive summary .....	iii
Sommaire .....	iv
Table of contents .....	v
List of figures .....	vii
List of tables .....	viii
Acknowledgements .....	x
1 Introduction.....	1
1.1 Background .....	1
1.2 Data sources .....	1
1.3 Data integrity .....	2
1.4 Report objectives .....	2
1.5 Past efforts.....	2
1.6 Document Structure.....	4
2 Methodology .....	6
2.1 Requirements and constraints .....	6
2.2 Desirable objectives and caveats .....	6
2.3 Criteria.....	6
2.4 Database solutions .....	7
2.5 Deliverables.....	8
2.6 Performance.....	8
3 Database design .....	9
3.1 Import workflow.....	10
3.2 Database schema .....	10
3.3 Database Dictionary .....	13
3.3.1 Ship.....	13
3.3.2 Class.....	14
3.3.3 ShipAction .....	14
3.3.4 ActionStation .....	15
3.3.5 Station.....	15
3.3.6 ActionVehicle.....	16
3.3.7 Vehicle.....	16
3.3.8 ShipActivity .....	17
3.3.9 ShipDmg .....	17
3.3.10 ShipHomeport.....	18
3.3.11 Homeport .....	19

3.3.12	ShipPosition.....	19
3.3.13	DataEntryFlag.....	20
3.3.14	PositionFlag .....	20
3.3.15	ShipReadiness.....	21
3.3.16	Readiness .....	21
3.3.17	ShipStatus .....	22
3.3.18	Status .....	22
3.3.19	ShipSun.....	23
3.3.20	ShipTimezone.....	23
3.3.21	ShipWeather .....	24
3.3.22	StemSuffix .....	25
3.3.23	Tally.....	26
4	Recommendations.....	28
5	Conclusion .....	30
	List of symbols/abbreviations/acronyms/initialisms .....	31
	References .....	32
Annex A	Normal Forms.....	34
A.1	First Normal Form.....	34
A.2	Second Normal Form .....	34
A.3	Third Normal Form .....	34
A.4	Fourth Normal Form.....	34
A.5	Fifth Normal Form .....	35
A.6	Sixth Normal Form.....	35
Annex B	Star and Snowflake Schema .....	36
B.1	Fact table .....	36
B.2	Dimension table.....	36
Annex C	Anchor Modeling.....	37
Annex D	Populating the database .....	40
D.1	Flat file format.....	40
D.2	SQL Master table.....	42
D.3	Database schema .....	43
D.4	Batch automation.....	43
D.5	Digitization .....	44

## List of figures

---

Figure 1: Database Schema .....	11
Figure 2: Relational Keys Connecting Database Tables .....	12
Figure 3: Ship table definition .....	13
Figure 4: Class table definition.....	14
Figure 5: ShipAction table definition .....	14
Figure 6: ActionStation .....	15
Figure 7: Station table definition .....	15
Figure 8: ActionVehicle .....	16
Figure 9: Vehicle table definition.....	16
Figure 10: ShipActivity table definition.....	17
Figure 11: ShipDmg table definition.....	18
Figure 12: ShipHomeport table definition.....	18
Figure 13: Homeport table definition .....	19
Figure 14: ShipPosition table definition.....	19
Figure 15: DataEntryFlag table definition.....	20
Figure 16: PositionFlag table definition .....	20
Figure 17: ShipReadiness table definition.....	21
Figure 18: Readiness table definition .....	21
Figure 19: ShipStatus table definition .....	22
Figure 20: Status table definition .....	22
Figure 21: ShipSun table definition.....	23
Figure 22: ShipTimezone table definition .....	24
Figure 23: ShipWeather table definition .....	25
Figure 24: StemSuffix table definition .....	26
Figure 25: Tally table definition.....	26
Figure B.1: Example of a Snowflake Schema [9] .....	36
Figure C.1: Anchor Model (See Reference [11]) .....	37
Figure D.1: Data Import Process .....	40
Figure D.2: Flat File Format Specification for Data Categories .....	41

## List of tables

---

Table 1: Modified Computational Decision Making Matrix (CDMM).....	8
Table 2: Ship table description .....	13
Table 3: Ship table example rows .....	13
Table 4: Class table description.....	14
Table 5: Class table example rows .....	14
Table 6: ShipAction table description .....	14
Table 7: ShipAction table example rows.....	14
Table 8: ActionStation table description .....	15
Table 9: ActionStation table example rows.....	15
Table 10: Station table description .....	15
Table 11: Station table example rows .....	15
Table 12: ActionVehicle table description .....	16
Table 13: ActionVehicle table example rows.....	16
Table 14: Vehicle table description .....	16
Table 15: Vehicle table example rows .....	16
Table 16: ShipActivity table description.....	17
Table 17: ShipActivity table example rows .....	17
Table 18: ShipDmg table description .....	17
Table 19: ShipHomeport table description.....	18
Table 20: ShipHomeport table example rows .....	18
Table 21: Homeport table description .....	19
Table 22: Homeport table example rows.....	19
Table 23: ShipPosition table description.....	19
Table 24: ShipPosition table example rows .....	20
Table 25: DataEntryFlag table description.....	20
Table 26: DataEntryFlag table example rows .....	20
Table 27: PositionFlag table description .....	20
Table 28: PositionFlag table example rows.....	21
Table 29: ShipReadiness table description.....	21
Table 30: ShipReadiness table example rows .....	21

Table 31: Readiness table description .....	21
Table 32: Readiness table example rows.....	22
Table 33: ShipStatus table description .....	22
Table 34: ShipStatus table example rows.....	22
Table 35: Status table description.....	22
Table 36: Status table example rows .....	23
Table 37: ShipTimezone table description .....	23
Table 38: ShipWeather table description.....	24
Table 39: StemSuffix table description .....	25
Table 40: StemSuffix table example rows.....	26
Table 41: Tally table description.....	26
Table 42: Tally table example rows .....	26

## Acknowledgements

---

Dr. Ramzi Mirshak's knowledge on past efforts for the Ship Activity and Location Database and extensive involvement in different analysis techniques in his past works provided invaluable assistance in the Naval Ship Database project. This knowledge of the Ship Activity and Location Database helped highlight flaws in past efforts allowing the Naval Ship Database to improve on past efforts. His knowledge of analysis techniques illustrated how the data can be used in future projects. As a result, the Naval Ship Database project was designed with usability in mind by allowing easy but fast and powerful methods in querying the data. Dr. Mirshak has also provided helpful comments and technical background to improve the quality of this report.

Mr. Paul Massel's professional experience as an officer in the Royal Canadian Navy was instrumental in understanding RCN jargon and common practices on a typical RCN vessel. This experience led to the development of several error detection methods to be used during the data import process, the design layout of the database schema, integral database constraints, and proper normalization techniques. As a result, the Naval Ship Database project was also designed around the logical organization and integrity of data improving the quality of the data to be stored in the database. Mr. Massel has also collected data pertaining to day to day activities on RCN vessels that has been imported into the Naval Ship Database project.

# 1 Introduction

---

## 1.1 Background

The field of Operational Research often benefits from the analysis of vast amounts of historical data. The Maritime Operational Research Team (MORT) has access to data collected from navy ships from the 1990s to present. To assist with current and upcoming projects, the Defense Research and Development Canada (DRDC) Center for Operational Research and Analysis (CORA) requires quick access and flexible storage of this data.

## 1.2 Data sources

This project combines existing data in the Royal Canadian Navy (RCN) files such as Operational Schedules and Ship Logs. This data has been compiled and held by MORT. Within the compilation process, the datasets listed below exist as Excel spreadsheets.

1. **Ship Position Data:** This dataset, originally described in Reference, contains the geo-location of a set of ships. This position and port data along with the ship's operational status spans from the beginning of 1990 to the end of 2009. References [3] and [2] detail the quality of the collected data and have additional flags detailing the level of quality for each data point collected.
2. **Ship Activity Data:** Although not yet officially documented, this unpublished dataset has been used in several studies by MORT in References [4] and [5]. The dataset details each ship's scheduled tasks on a day to day basis, readiness status, and homeport information.
3. **Distance Made Good (DMG) Data:** This data can be found in the ship log data within library archives. This data shows the distance travelled broken down into each watch of the day starting from the Middle watch and looping around to the First watch. CAE Standing Offer Task 131 in Reference [2], an external contract with CAE allowed this information to be collected from ship logs.
4. **Weather Data:** Weather conditions recorded include wind conditions, sea state, cloud cover, barometric pressure, and visibility conditions. The data are currently being collected by CAE under the same contract as the DMG data.
5. **Ship Log Entries:** These ship logs are currently examined with key entries copied into an Excel spreadsheet by an external contractor. These ship logs include specific actions recorded by the officer of the watch such as vehicle launches, RAS details, and station calls. Past data collections efforts help provide a small dataset to analyze. More data are currently being collected by CAE under the same contract as the DMG data to provide studies to support PMO CSC (See Reference [1]).

Similar to the findings in past works relating to the quality of position data (See Reference [3]), data integrity issues exist even at the lowest level in the ship logs. In spite of collection and

storage efforts, translation errors, invalid values, and contradicting data values still exist in the currently stored Excel form.

### **1.3 Data integrity**

Several different data integrity issues were encountered when analyzing the data. One such issue is missing data values. For example in ship actions, a vehicle would be recorded as launched, but never recovered. Another issue is that key identifiers of the data values have discrepancies. For instance, ship activities are recorded in a date-duration format, but there are a few days in the activity dataset with multiple overlapping activities when each ship is supposed to only have one designated activity per day. On top of all these issues, the data values themselves are erroneous; impossible to detect, but can ultimately skew the results during the analysis. These errors include spelling mistakes and incorrect values which are both more than likely due to errors in translation. These erroneous values can be identified by outliers identified using the interpolation of adjacent data values. As a result, values with discrepancies will have to be corrected or excluded when transferring to a new method of storage.

### **1.4 Report objectives**

The objectives of this report are to document the:

1. Datasets and provide advice on the best way to combine them into a database
2. Recommended software tools to support the development and use of a database
3. Database schema and a related data dictionary
4. Construction and population of the database with provided datasets
5. Construction of standard queries and reports
6. Resolution of a number of issues in current datasets, including inconsistent port name spelling and inconsistent port name scheme
7. Provision of a written report that summarizes the work in this project

### **1.5 Past efforts**

A solution from CAE Professional Services has attempted to import the data into a Ship Activity and Location Database (Reference [2]). It uses SQL Server to import and store the data from the position and activity Excel datasets. Unfortunately, this was not optimal due to multiple issues residing with the database that was provided by CAE. In addition, the database design associated with this effort had several flaws. That said, some of its core design ideas can be used for designing a new solution.



Data integrity issues such as contradicting values on specific days, conflicted overlapping date ranges, and correctable erroneous data values existed in the final database delivered by CAE. Identifying them without prior in-depth exposure to the data or well established table constraints would have proved to be a very challenging task. Data integrity issues would have only worsened if the data was imported using techniques that resulted in irreversible data corruption. Additionally, almost all primary keys were changed from their original values. At times, these values were not documented or transferred to the database from their original data source. As a result, some key identifiers in helping to understand and correct the values may have been lost during the import process.

The provided database schema designed by CAE shows the existence of redundant tables and columns. One such example is that each table has a “PrimaryKey” column which is an arbitrary integer attempting to emulate the use of a globally unique identifier (GUID). It is designated as the primary key of the table and sometimes foreign keys in other tables. This method of choosing the primary key ignores the existing data which, in cases such as activity codes, could have been used as the primary key instead. While the solution theoretically complies with the First Normal Form (1NF) (See Annex A for Database Normalization), it does not fully satisfy its purpose. The result can be seen as the data integrity issues described above. The PrimaryKey column should be replaced with a primary key consisting of the ship identifier and date to enforce dependency on the primary key. This allows row data values or columns that reference foreign keys to depend on one constrained key. At the same time, old mapped data still exists in table columns. Even if the mapping was well defined, the data existed in a form breaking other normalization standards. The database fails to adhere to the Second Normal Form (2NF) or Third Normal Form (3NF) in that the data values have no relation to the primary key other than ensuring that there are no duplicate rows. The database fails to adhere to industry standard database normalization techniques. The solution is more error prone as erroneous data can be inserted into the tables without the operator’s knowledge.

On a larger scope, the database also contains redundant tables. For instance, the Ship and Class tables have ShipClass as an intermediate relationship table storing the relationship between ship and classes of ships. The fundamental object for relational purposes in this database is a single RCN ship. A table is only required if there is a many-to-one (m:1) relation to another table in the database. In this case, the problem can be simplified by assigning the most current classification to the ship making the intermediate table redundant.

Several fundamental design flaws exist in the database presented in Reference [2]:

1. The solution’s data structure stores dates as date-durations entries in attempt to minimize data storage. However, most of the data sources with exception to the Ship Activity dataset include date-value entries. Using date-durations data structure requires a grouping method be specified by the data source during the data import process. For instance, activities must be grouped by the same activity span and port status must be grouped by the same port status span. Data mutate operations are more likely to be erroneous as opposed to being imported straight from the data source. As well, this leads to data integrity loopholes as there is no database level constraint to check for overlapping date-duration. In addition, not all data can be grouped into date-duration entries – consider positional data – so it does not make logical sense to organize the data in this manner.

2. Other design flaws include the lack of standardization in table column names. A remark for the row would sometimes be referred to as description, remarks, meaning, name, or an object's name (which should in theory be reserved for table names). In general, this type of nomenclature is confusing for database design and obfuscates the schema for the operator. Again the database fails to adhere to industry standards when it comes to database naming conventions. In addition, the absence of a Tally Table will make it difficult for a database operator to script queries to analyze the data; and
3. The existing database is simply not flexible enough for MORT's analysis requirements for some of its projects. SQL databases are fundamentally optimized for JOIN operations which inherently combine multiple datasets and output them into useful table results. To aggregate or correlate data stored in the database, values are best stored as data points as opposed to ranges.

In addition to the concerns listed above, MORT has a growing need for data collection and storage. It is foreseen in upcoming projects (See Reference [1]), new data from FFH and DDH ships will highlight different events occurring on the ship in the ship logs. Data such as weather and daily activities will not fit the existing schema defined in Reference [2] as the data may not correspond to the dates provided in existing tables. The existing schema can be expanded and new tables can be added. However, factoring the flaws of the existing CAE solution into the problem, it was determined that it would be better to redesign the schema with the intent to assert data integrity across the rows and tables in a more robust and conventional manner.

## **1.6 Document Structure**

Section 2 describes the methodology in determining the solution for the project and highlights database design decisions to ensure that desirable objectives are met. Section 3 primarily focuses on the database schema definition. Section 4 provides recommendations to build on top of the project. Section 5 provides conclusions for the project. Appendices are attached as Annexes following the end of the report so as to provide detailed references on report topics for further reading.



## **2 Methodology**

---

### **2.1 Requirements and constraints**

The project solution must meet the requirements and must not exceed constraints of the project scope. The solution must adhere to the following requirements and constraints:

1. Any software used must comply with DND security policies;
2. Data storage techniques must adhere to 1NF and 2NF standards;
3. The solution must implement constraints to reject erroneous data; and
4. Implementation time cannot exceed time requirements.

### **2.2 Desirable objectives and caveats**

The objective of this project is to put these datasets into a storage form that can be easily analyzed later. The Ship Activity and Location Database will be replaced with a new solution to improve upon its design flaws and build upon the core ideas from its schema. The new solution should have the following desirable objectives and caveats:

1. The database should deal with missing data, key identifier errors, and erroneous values;
2. The data should be stored in a MS SQL Server;
3. The database should be easily searchable;
4. New features for analysis should be easy to add;
5. New features should easily integrate into old ones;
6. Data should be easy to import from Excel files; and
7. Data storage techniques should adhere to 3NF standards.

### **2.3 Criteria**

Well labeled criteria will help determine the migration costs, project usability, and future maintenance costs of the solution. They will help highlight the functionality of each solution and are all equally weighted as they all help contribute to the success of future projects. The fulfilment of each criteria category is measured from 0 to 4, with 0 being the lowest and 4 being the highest, and will be scored by the sub-points to which the solutions map. The solution should have the best overall score on the following criteria.

1. Availability of the software to CORA (Availability)

- a. Approval Status
    - i. Would be easily approved on the DREnet (1)
    - ii. Already approved for the DREnet (2)
  - b. Industry Usage
    - i. Is somewhat common in industry standards (1)
    - ii. Is highly common in industry standards (2)
- 2. Data integrity with the use of constraints (Robustness)
  - a. Robustness to erroneous data entries
    - i. Has built-in data constraints (4)
- 3. Ability to query existing data (Usability)
  - a. Scripting Standard
    - i. Programmatic loop (1)
    - ii. Query builder (2)
    - iii. Complies to SQL Standard (3)
    - iv. Complies to SQL Standard and has powerful built-in functions (4)
- 4. Flexible to import data (Flexibility)
  - a. Import
    - i. Programmatic parse and sort data (1)
    - ii. Manually import new data (2)
    - iii. Script to import indirectly into tables (3)
    - iv. Script to import directly into tables (4)

## 2.4 Database solutions

The Ship Activity and Location Database as documented in Reference [2] originally considers MS Access, Django, and a Standalone SQLite Database. The report states that MS Access was recommended but selected SQL Server as the final database storage solution. Adding the commonly used MySQL data storage solution, the options will be lightly considered once again.

The assessment of database solution options is displayed as a modified Computational Decision Making Matrix (CDMM) in Table 1. As well, it is assumed to be trivial that keeping the data in CSV or Excel format is unfeasible.

*Table 1: Modified Computational Decision Making Matrix (CDMM)*

Criteria	MS Access	Django	SQLite	SQL Server	MySQL
Criteria 1 (Availability)	2	0	2	4	3
Criteria 2 (Robustness)	2	1	4	4	4
Criteria 3 (Querying Ability)	2	2	3	4	3
Criteria 4 (Flexibility)	2	1	3	3	3
<b>Total</b>	8	4	12	15	13

From the Computational Decision Making Matrix, it is clear that SQL Server should be selected as the database management system (DBMS).

## 2.5 Deliverables

Upon completion of the project, the following deliverables are required:

1. Database with imported data
2. Relevant SQL import scripts
3. Relevant SQL data analysis scripts
4. A Technical Note (TN) detailing the design, implementation, and schema of the database

## 2.6 Performance

The SQL standard is advantageous in performance when using data formatted into tables. SQL Server is optimized for JOIN operations which allow multiple tables to be combined

### 3 Database design

---

Several steps were taken to ensure that the database design improved upon past efforts. The first step is to identify the sources of data and then find a way to organize those data into consistent and usable tables for easy querying. The data sources in each Excel spreadsheet were classified as major categories, with the data contained in each spreadsheet broken down and can be grouped into different tables as follows:

1. **Actions** – daily events that occur in the ship’s daily logs
  - a. Actions: An event recorded in the ship entry logs (eg. vehicle launch)
  - b. Path / Port: Which port the ship is docked at, originated from, and has as its destination to, if any
2. **Activities** – day to day events that the ship is tasked with
  - a. Activity: The ship’s purpose to be in its current position (eg. transit, operation, equipment trials, patrol)
  - b. Readiness: The force generation cycle status
  - c. Homeport: The ship’s homeport in Canada
3. **Position** – coordinate and status relating to geo-location of a ship
  - a. Position: Longitude and latitude coordinates
  - b. Time: Time of day for specified position; defaults to noon if unspecified
  - c. Status: The ship’s dock status (eg. anchored, moored to buoy)
  - d. Port: Which port the ship is docked at
4. **Weather** – operating conditions and distance made good performance on a ship
  - a. Weather:
  - b. Time Zone: Coordinated Universal Time (UTC) time zone code
  - c. Distance Made Good (DMG): Distance made good broken down for each watch
  - d. Path / Port: Docked port, origin port, and destination port

In addition to individual feature column data within these categories, the database can use common normalization techniques described in Annex A to enhance the integrity and standardize

the dataset. The result allows for three classifications of tables as described in the Star and Snowflake Schema (See Annex B).

1. **Dimension Table** – a table used to hold instances of objects of a specific type defining a column in another table with its attributes typically in a 1:1 relationship (eg. Ship, Class, Vehicle, Station, etc.)
2. **Fact Table** – basic structure to hold data regarding objects in a relational database with some of its attributes typically in a m:1 relationship (eg. ShipPosition, ShipAction, ShipActivity, etc.)
3. **Auxiliary Table** – tables used to aid in performance and ease of creation of queries (eg. Tally table)

### 3.1 Import workflow

The data exists in its most raw form as written entries in ship logs. These ship logs are currently scanned into PDF form or parsed into an excel spreadsheet format by CAE Professional Services. Within the Department of Defense (DND), the data exists in Excel and Comma Separated Values (CSV) format.

### 3.2 Database schema

The schema is the organization of data to create a blueprint of how a database is divided into database tables. It helps to visualize the organization of the data as relational tables and respective columns.

To design the schema, it should be noted that feature data can be sparse in the sense that some data may not be available for some dates. Data across several features may overlap on some dates but may not be available for others. The data being stored is also temporal, which means that that data values each have a corresponding ship and date. Sometimes the data may include a time as well.

To highlight dimensions in the features, a Snowflake Schema (see Annex B) was selected to help normalize data into its 3NF. The database design schema starts at the most central object in the database, a ship. Branching off the Ship table are relational tables containing the temporal data. ShipAction and ShipPosition are large Fact Tables that contain useful data. The schema for the relational database is shown in Figure 1.



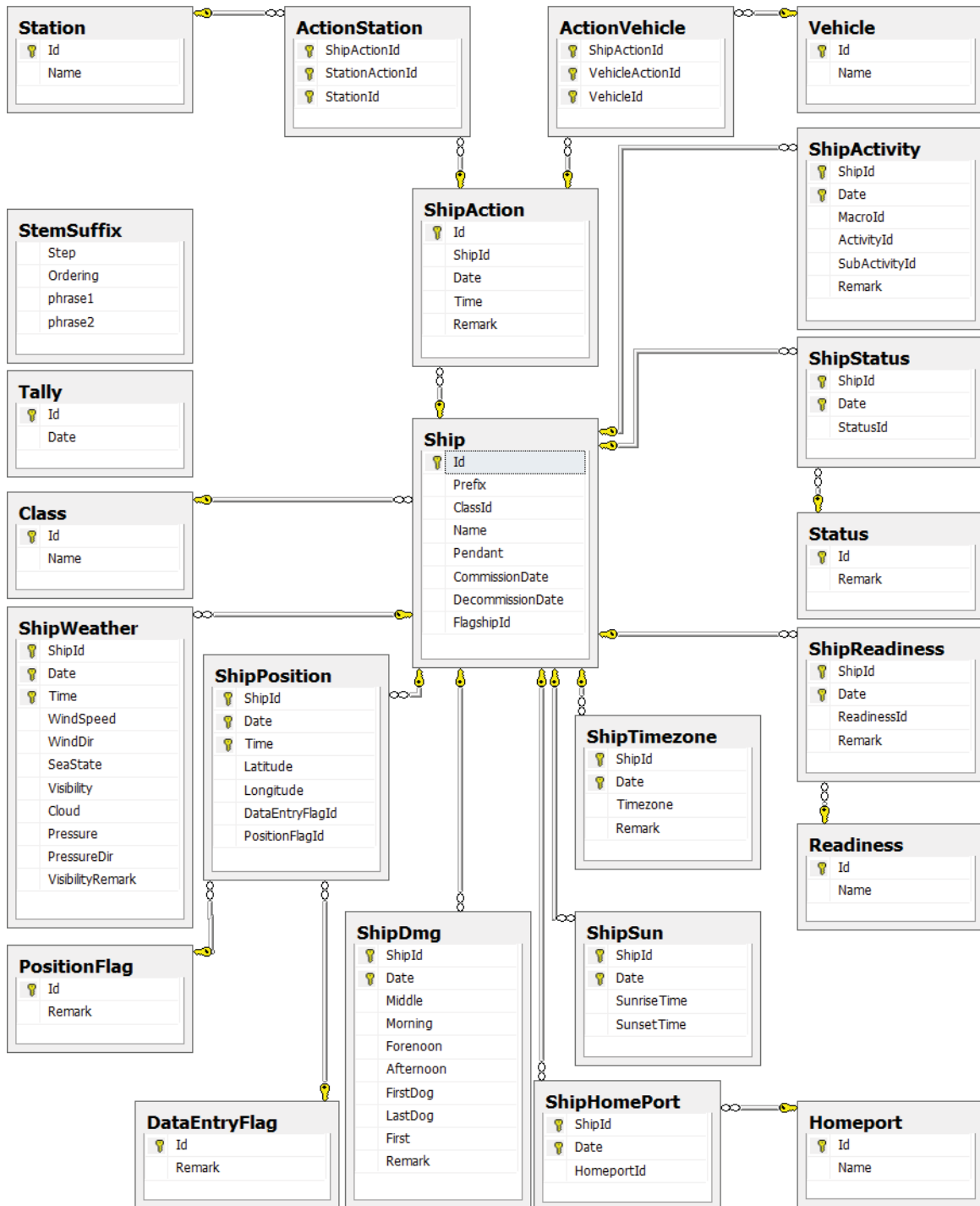


Figure 1: Database Schema

The Primary Key constraint uniquely identifies each record in a database table. By using a Foreign Key constraint on specific columns, values in these constrained columns can refer to a

primary key of another table. The relational keys allow database tables to be connected by defining explicit relations between table columns and help to enforce referential integrity constraints as seen in Figure 2. Thus, database tables are connected by defining these explicit relations between table columns and help to enforce referential integrity constraints. To fully understand the relation, non-primary key columns were stripped from Figure 1 and only primary keys exist in Figure 2.

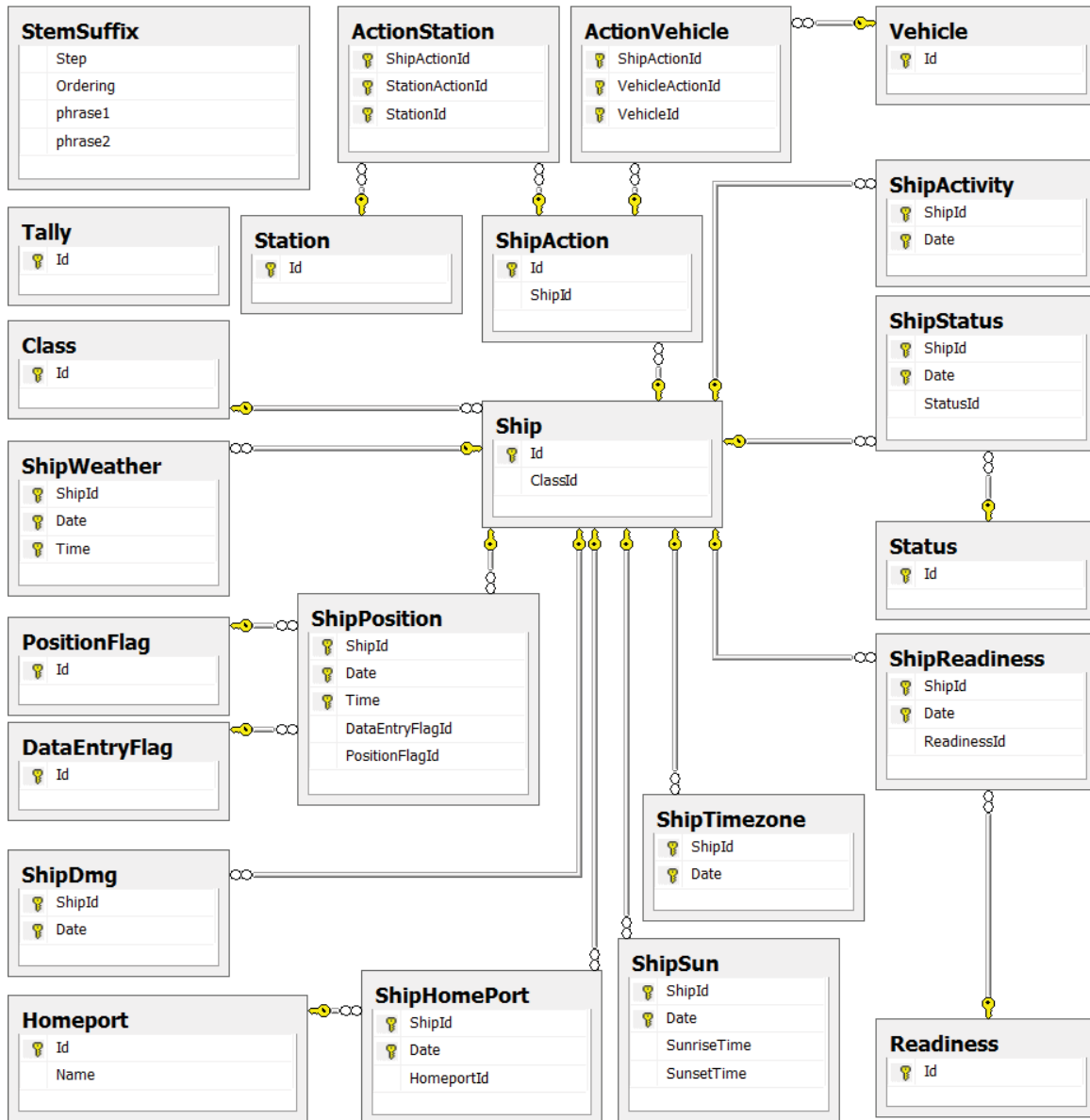


Figure 2: Relational Keys Connecting Database Tables


### 3.3 Database Dictionary

This section provides a definition of the database table columns, data types, and provides a few examples of values that can be contained within the table.

#### 3.3.1 Ship

*Table 2: Ship table description*

Column Name	Description
<b>Id</b>	Ship identifier
<b>Prefix</b>	Ship prefix namesake
<b>ClassId</b>	Class identifier
<b>Name</b>	Ship name
<b>Pendant</b>	Ship pendant
<b>CommissionDate</b>	Ship commission date
<b>DecommissionDate</b>	Ship decommission date; NULL if still active
<b>FlagshipId</b>	Ship Id of the ship

	Column Name	Data Type	Allow Nulls
	<b>Id</b>	int	<input type="checkbox"/>
	Prefix	varchar(10)	<input type="checkbox"/>
	ClassId	int	<input type="checkbox"/>
	Name	varchar(100)	<input type="checkbox"/>
	Pendant	int	<input type="checkbox"/>
	CommissionDate	date	<input type="checkbox"/>
	DecommissionDate	date	<input checked="" type="checkbox"/>
	FlagshipId	int	<input type="checkbox"/>
			<input type="checkbox"/>

*Figure 3: Ship table definition*

*Table 3: Ship table example rows*

Id	Prefix	ClassId	Name	Pendant	CommissionDate	DecommissionDate	FlagshipId
1	HMCS	1	Preserver	510	1970-08-07	NULL	1
2	HMCS	1	Protecteur	509	1969-08-30	NULL	2
3	HMCS	1	Provider	508	1963-08-28	NULL	2
4	HMCS	2	Gatineau	236	1959-02-17	1993-05-24	8

### 3.3.2 Class

Table 4: Class table description

Column Name	Description
<b>Id</b>	Class identifier
<b>Name</b>	Class name


	Column Name	Data Type	Allow Nulls
	<b>Id</b>	int	<input type="checkbox"/>
	<b>Name</b>	varchar(20)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 4: Class table definition

Table 5: Class table example rows

Id	Name
1	AOR
2	DDE

### 3.3.3 ShipAction

Table 6: ShipAction table description

Column Name	Description
<b>Id</b>	Action identifier
<b>ShipId</b>	Ship identifier
<b>Date</b>	Action date
<b>Time</b>	Action time
<b>Remark</b>	Action entry log


	Column Name	Data Type	Allow Nulls
	<b>Id</b>	int	<input type="checkbox"/>
	<b>ShipId</b>	int	<input type="checkbox"/>
	<b>Date</b>	date	<input type="checkbox"/>
	<b>Time</b>	time(7)	<input type="checkbox"/>
	<b>Remark</b>	varchar(1000)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 5: ShipAction table definition

Table 7: ShipAction table example rows

Id	ShipId	Date	Time
1	1	2001-10-18	01:35:00.0000000
2	1	2001-10-18	01:48:00.0000000

### 3.3.4 ActionStation

Table 8: ActionStation table description

Column Name	Description
ShipActionId	Action identifier
StationActionId	Station action identifier
StationId	Station identifier




	Column Name	Data Type	Allow Nulls
	ShipActionId	int	<input type="checkbox"/>
	StationActionId	int	<input type="checkbox"/>
	StationId	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 6: ActionStation

Table 9: ActionStation table example rows

ShipActionId	StationActionId	StationId
8	1	4
14	2	4

### 3.3.5 Station

Table 10: Station table description

Column Name	Description
Id	Station identifier
Name	Station name


	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(20)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 7: Station table definition

Table 11: Station table example rows

Id	Remark
0	Unknown
1	Rescue

### 3.3.6 ActionVehicle

Table 12: ActionVehicle table description

Column Name	Description
ShipActionId	Action identifier
VehicleActionId	Vehicle action identifier
VehicleId	Station identifier

	Column Name	Data Type	Allow Nulls
►	ShipActionId	int	<input type="checkbox"/>
►	VehicleActionId	int	<input type="checkbox"/>
►	VehicleId	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 8: ActionVehicle

Table 13: ActionVehicle table example rows

ShipActionId	VehicleActionId	VehicleId
2	0	1
3	1	1

### 3.3.7 Vehicle

Table 14: Vehicle table description

Column Name	Description
Id	Vehicle identifier
Name	Vehicle name

	Column Name	Data Type	Allow Nulls
►	Id	int	<input type="checkbox"/>
	Name	varchar(20)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 9: Vehicle table definition

Table 15: Vehicle table example rows

Id	Remark
0	Unknown
1	RIB

### 3.3.8 ShipActivity

Table 16: ShipActivity table description

Column Name	Description
ShipId	Ship identifier
Date	Activity date
MacroId	Macro activity identifier
ActivityId	Activity identifier
SubActivityId	Sub-activity identifier
Remark	Activity remarks



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	MacroId	int	<input checked="" type="checkbox"/>
	ActivityId	int	<input checked="" type="checkbox"/>
	SubActivityId	int	<input checked="" type="checkbox"/>
	Remark	varchar(100)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 10: ShipActivity table definition

Table 17: ShipActivity table example rows

ShipId	Date	MacroId	ActivityId	SubActivityId	Remark
1	1990-01-11	NULL	600	NULL	smp
1	1990-01-12	NULL	600	NULL	smp

### 3.3.9 ShipDmg

Table 18: ShipDmg table description

Column Name	Description
ShipId	Ship identifier
Date	DMG date
Middle	Middle watch DMG (NM)
Morning	Morning watch DMG (NM)
Forenoon	Forenoon watch DMG (NM)
Afternoon	Afternoon watch DMG (NM)
FirstDog	First Dog watch DMG (NM)
LastDog	Last Dog watch DMG (NM)
First	First watch DMG (NM)
Remark	DMG remarks



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	First	int	<input type="checkbox"/>
	Middle	int	<input type="checkbox"/>
	Morning	int	<input type="checkbox"/>
	Forenoon	int	<input type="checkbox"/>
	Afternoon	int	<input type="checkbox"/>
	FirstDog	int	<input type="checkbox"/>
	LastDog	int	<input type="checkbox"/>
	Remark	varchar(100)	<input type="checkbox"/>

Figure 11: ShipDmg table definition

ShipId	Date	Middle	Morning	Forenoon	Afternoon	FirstDog	LastDog	First	Remark
1	2001-10-17	0	0	0	0	5	10	20	Operation Apollo
1	2001-10-18	25	22	38	29	18	17	39	Operation Apollo

### 3.3.10 ShipHomeport

Table 19: ShipHomeport table description

Column Name	Description
ShipId	Ship identifier
Date	Homeport date
HomeportId	Homeport identifier



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	HomeportId	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 12: ShipHomeport table definition

Table 20: ShipHomeport table example rows

ShipId	Date	Homeport
1	1990-01-11	CFB Halifax
1	1990-01-12	CFB Halifax



### 3.3.11 Homeport

Table 21: Homeport table description

Column Name	Description
<b>Id</b>	Homeport identifier
<b>Name</b>	Homeport name

Column Name	Data Type	Allow Nulls
<b>Id</b>	int	<input type="checkbox"/>
<b>Name</b>	varchar(20)	<input type="checkbox"/>

Figure 13: Homeport table definition

Table 22: Homeport table example rows

Id	Name
1	CFB Halifax
2	CFB Esquimalt

### 3.3.12 ShipPosition

Table 23: ShipPosition table description

Column Name	Description
<b>ShipId</b>	Ship identifier
<b>Date</b>	Position date
<b>Time</b>	Position time
<b>Latitude</b>	Position latitude
<b>Longitude</b>	Position longitude
<b>DataEntryFlagId</b>	Data entry flag identifier
<b>PositionFlagId</b>	Position flag identifier

<b>ShipId</b>	int	<input type="checkbox"/>
<b>Date</b>	date	<input type="checkbox"/>
<b>Time</b>	time(7)	<input type="checkbox"/>
<b>Latitude</b>	decimal(7, 4)	<input type="checkbox"/>
<b>Longitude</b>	decimal(7, 4)	<input type="checkbox"/>
<b>DataEntryFlagId</b>	int	<input type="checkbox"/>
<b>PositionFlagId</b>	int	<input type="checkbox"/>

Figure 14: ShipPosition table definition

Table 24: ShipPosition table example rows

ShipId	Date	Time	Latitude	Longitude	DataEntryFlagId	PositionFlagId
1	1998-04-17	12:00:00.0000000	44.5083	-60.5167	0	0
1	1998-04-21	12:00:00.0000000	44.6400	-63.9600	0	0

### 3.3.13 DataEntryFlag

Table 25: DataEntryFlag table description

Column Name	Description
<b>Id</b>	Data entry flag identifier
<b>Remark</b>	Data entry flag remarks


	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Remark	varchar(100)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 15: DataEntryFlag table definition

Table 26: DataEntryFlag table example rows

Id	Remark
0	From log
1	Interpolated

### 3.3.14 PositionFlag

Table 27: PositionFlag table description

Column Name	Description
<b>Id</b>	Position flag identifier
<b>Remark</b>	Position flag remarks


	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Remark	varchar(100)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 16: PositionFlag table definition

Table 28: PositionFlag table example rows

Id	Remark
0	No integrity issues detected
1	Abnormal speed detected (25 kts)

### 3.3.15 ShipReadiness

Table 29: ShipReadiness table description

Column Name	Description
ShipId	Ship identifier
Date	Readiness date
ReadinessId	Readiness identifier
Remark	Readiness remark



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	ReadinessId	int	<input type="checkbox"/>
	Remark	varchar(10)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 17: ShipReadiness table definition

Table 30: ShipReadiness table example rows

ShipId	Date	ReadinessId	Remark
1	1998-04-01	6	N
1	1998-04-02	6	N

### 3.3.16 Readiness

Table 31: Readiness table description

Column Name	Description
Id	Readiness identifier
Name	Readiness name


	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(20)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 18: Readiness table definition

Table 32: Readiness table example rows

Id	Name
0	Unknown
1	Out of routine

### 3.3.17 ShipStatus

Table 33: ShipStatus table description

Column Name	Description
ShipId	Ship identifier
Date	Status date
StatusId	Status identifier



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	StatusId	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 19: ShipStatus table definition

Table 34: ShipStatus table example rows

ShipId	Date	StatusId
1	1996-08-01	6
1	1996-08-02	6

### 3.3.18 Status

Table 35: Status table description

Column Name	Description
ShipId	Status identifier
Remark	Status remark


	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Remark	varchar(100)	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 20: Status table definition

Table 36: Status table example rows

Id	Remark
1	Ship in port
2	Ship on day trip

### 3.3.19 ShipSun

Table 37: ShipSun table description

Column Name	Description
ShipId	Ship identifier
Date	Record time
SunriseTime	Sunrise time
SunsetTime	Sunset time



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	StatusId	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 21: ShipSun table definition

Table 38: ShipSun table example rows

ShipId	Date	SunriseTime	SunsetTime
23	2008-01-01	08:05:00.0000000	16:29:00.0000000
23	2008-01-02	08:05:00.0000000	16:30:00.0000000

### 3.3.20 ShipTimezone

Table 39: ShipTimezone table description

Column Name	Description
ShipId	Ship identifier
Date	Timezone date
Timezone	Timezone offset (UTC)
Sunrise	Sunrise time
Sunset	Sunset time
Remark	Timezone remark



	Column Name	Data Type	Allow Nulls
	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	Timezone	int	<input checked="" type="checkbox"/>
	Sunrise	time(7)	<input checked="" type="checkbox"/>
	Sunset	time(7)	<input checked="" type="checkbox"/>
	Remark	nvarchar(100)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 22: ShipTimezone table definition

Table 40: ShipTimezone table example rows

ShipId	Date	Timezone	Remark
1	2008-04-25	-4	Q
1	2008-05-11	-4	Q

### 3.3.21 ShipWeather

Table 41: ShipWeather table description

Column Name	Description
ShipId	Ship identifier
Date	Weather date
Time	Weather time
WindSpeed	Wind speed (m/s)
WindDir	Wind direction (bearing)
SeaState	Sea state (1-8)
Visibility	Visibility (Nautical Miles; 999 is unrestricted)
Cloud	Cloud cover
Pressure	Barometric Pressure (Millibars)
PressureDir	Barometric Pressure change direction (1 inc / 0 stable / -1 dec)
VisibilityRemark	Visibility remarks

	Column Name	Data Type	Allow Nulls
PK	ShipId	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	Time	time(7)	<input type="checkbox"/>
	WindSpeed	int	<input checked="" type="checkbox"/>
	WindDir	int	<input checked="" type="checkbox"/>
	SeaState	int	<input checked="" type="checkbox"/>
	Visibility	decimal(6, 3)	<input checked="" type="checkbox"/>
	Cloud	int	<input checked="" type="checkbox"/>
	Pressure	decimal(6, 2)	<input checked="" type="checkbox"/>
	PressureDir	int	<input checked="" type="checkbox"/>
	VisibilityRemark	nvarchar(100)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 23: ShipWeather table definition

ShipId	Date	Time	WindSpeed	WindDir	SeaState	Visibility	Cloud	Pressure	PressureDir	VisibilityRemark
1	2008-02-04	13:40:00.0000000	8	300	1	12.000	2	1020.00	0	12
1	2008-02-04	14:57:00.0000000	8	320	2	12.000	2	1022.00	1	Unres

### 3.3.22 StemSuffix

The StemSuffix table is an auxiliary table used to store suffix mappings for use with the Porter Stemmer algorithm. The Porter stemming algorithm (or ‘Porter stemmer’) is a process for removing the common suffixes to find the root word in English. Its use in the Naval Ship Database is during a normalization process where the ship log entries are analyzed for keywords implying certain actions (eg. launching a RIB).

Table 42: StemSuffix table description

Column Name	Description
Step	Stemmer algorithm step
Date	Stemmer algorithm ordering in step
phrase1	Search string
phrase2	Replace string

	Column Name	Data Type	Allow Nulls
▶	Step	int	<input type="checkbox"/>
	Ordering	int	<input type="checkbox"/>
	phrase1	nvarchar(15)	<input type="checkbox"/>
	phrase2	nvarchar(15)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 24: StemSuffix table definition

Table 43: StemSuffix table example rows

Step	Ordering	phrase1
1	0	sses
1	1	ies

### 3.3.23 Tally

The Tally table is an auxiliary table used to store generated sequences to improve performance of some queries.

Table 44: Tally table description

Column Name	Description
<b>Id</b>	Integer sequence
<b>Date</b>	Date sequence

	Column Name	Data Type	Allow Nulls
▶	Id	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 25: Tally table definition

Table 45: Tally table example rows

Id	Date
1	1970-01-01
2	1970-01-02





## 4 Recommendations

---

The current implementation of the Naval Ship Database can be built upon and improved. The Naval Ship Database project may be improved by further enhancing the normalization level. The database currently satisfies 3NF when using standard database normalization techniques. However the database can be further normalized up to the 6<sup>th</sup> Normal Form (6NF). Data storage requirements would be optimal for the specific schema with a 6NF implementation. However, a 6NF implementation is not only difficult to design manually, but queries are generally more obfuscated. It is possible to auto-generate the schema by using Anchor Modeling techniques (see Annex C).

The data can be further analyzed using advanced techniques. While the data can be queried in SQL, advanced analysis of the data can also be done in Matlab. Thus a connection method can be built between Matlab and SQL Server. Matlab offers several advantages for plotting visualization and advanced tools for analysis. For instance, Matlab has toolkits for plotting, regression fitting, cluster analysis, and neural networks. In addition, Matlab also handles data in the form of Matrices allowing for advanced parallel processing techniques for numeric computation if required. A bridged connection will allow computation analysis to be done in Matlab and allow the processed data to be imported back into SQL server. The data import workflow is how new data can be imported (See Annex D).

The data processing and analysis can be improved using automation methods. It is possible to explore machine learning alternatives. Machine learning can be used for data collection, automatic classification of existing data, and correlation between existing datasets. For instance, machine learning can be applied to analyze ship logs and digitize the documents into text. Additionally, machine learning can also be used to extract inferences from the data. In this case, analyzing the ship logs can be a classification problem where events such as helicopter launches can be classified into one group. Additionally, machine learning can also help improve data integrity by identifying duplicate entries or missing data. An example is when a helicopter is recovered and flying stations were raised twice but no such record of a launch was ever recorded. In this case, the machine learning algorithm can infer that the helicopter was launched when the first flying station was raised. Supervised training can be done with large datasets preformatted in SQL or incremental datasets; both methods improving the quality of the process output over time. It is also possible for unsupervised training methods to also help by making inferences using data collected from various sources. Whichever the case is, machine learning applications should be explored with this large dataset.

This page intentionally left blank.

## 5 Conclusion

---

The newly developed Naval Ship Database allows for current data to be imported from multiple datasets and stored into a SQL Server database. The database is flexible for further expansion when new features are required for analysis. Improving upon the design flaws and integrity issues of past efforts, the database is robust to new extensions allowing users to analyze the collected data.

## List of symbols/abbreviations/acronyms/initialisms

---

1:1	One-to-One
1:m	One-to-Many
M:m	Many-to-Many
1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
4NF	Fourth Normal Form
5NF	Fifth Normal Form
6NF	Sixth Normal Form
AOR	Auxiliary Oiler Replenishment
CAE	CAE Professional Services Inc.
CORA	Center for Operational Research and Analysis
CDMM	Computational Decision Making Matrix
CSV	Comma Separated Values
DBMS	Database Management System
DMG	Distance Made Good
DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
DREnet	Defense Research Network
GUI	Graphical User Interface
HMCS	His/Her Majesty's Canadian Ship
Kts	Knots (Speed)
MORT	Maritime Operational Research Team
NM	Nautical Mile
TN	Technical Note
R&D	Research & Development
RCN	Royal Canadian Navy
SQL	Structured Query Language
SQLite	Structured Query Language Lite
UTC	Coordinated Universal Time

## References

---

- [1] MARCORD 02-03, "Request for DRDC Support, Ship Position Analysis Task," DRMS DSR Serial # 2013-2, September 2009.
- [2] Robinson, Edward, "Ship Activity and Location Database," CAE Contractor Report, DRDC CORA CR 2012-256, October 2012.
- [3] Mirshak, Ramzi, "Quality Control of the Noon Ship Position Data Collection Study," DRDC CORA TN 2012-155, June 2012.
- [4] Mirshak, Ramzi, Massel, Paul, "Canadian Fleet Global Deployment Study," DRDC CORA TM 2013-015, February 2013.
- [5] Mirshak, Ramzi, "Helicopter and Boat Launching Activities on HMCS Protecteur During Operation Apollo," DRDC CORA LR 2012-177, August 2012.
- [6] Codd, Edgar Frank, A Relational Model of Data for Large Shared Data Banks, June 1970.
- [7] Date, Chris J., Darwen, Hugh, and Lorentzos, Nikos A., Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and Relation Theory to the Problem of Temporal Database Management, Oxford: Elsevier LTD, January 2003.
- [8] Pro Business Systems LLC, "Fact Table and Star Schema," 2012. [Online]. Available: [http://www.learndatamodeling.com/fact.php#.UcsGl\\_nVDgw](http://www.learndatamodeling.com/fact.php#.UcsGl_nVDgw). [Accessed June 2013].
- [9] "Star Schema," [Online]. Available: <http://s3.amazonaws.com/answer-board-image/200859514456334590688596725003307.jpg>. [Accessed August 2013].
- [10] Rönnbäck, Lars, "About Anchor Modeling," Anchor, [Online]. Available: [http://www.anchor modeling.com/?page\\_id=2](http://www.anchor modeling.com/?page_id=2). [Accessed June 2013].
- [11] Rönnbäck, Lars, "Anchor Modeling Example," 16 February 2011. [Online]. Available: [http://en.wikipedia.org/wiki/File:Anchor\\_Modeling\\_Example.svg](http://en.wikipedia.org/wiki/File:Anchor_Modeling_Example.svg). [Accessed June 2013].
- [12] Pinal Dave, "Load Comma Delimited File Into SQL Server," SQL Authority, 6 February 2008. [Online]. Available: <http://blog.sqlauthority.com/2008/02/06/sql-server-import-csv-file-into-sql-server-using-bulk-insert-load-comma-delimited-file-into-sql-server/>. [Accessed June 2013].



## **Annex A    Normal Forms**

---

Edgar F. Codd originally defined the first three Normal Forms in his work “A Relational Model of Data for Large Data Banks” back in 1970 (See Reference [6]). The Third Normal Form (3NF) was redefined as the Boyce-Codd Normal Form (BCNF) to enforce key requirements as well as directionality of column dependencies. [7]

The database schema designed in the Naval Ship Database is normalized to the Third Normal Form (3NF) as stated in the requirements. There are six Normal Forms theorized in Database theory, but the requirements for this project only specify 3NF as the higher order Normal Forms highly increase the complexity of manually designing the database as well as the ability to make powerful short queries. For instance, 6NF is known to lead to an explosion of tables exponentially increasing the complexity of the database. As such, there is an acceptable performance loss when factoring increased design complexity, likeliness of operator error, and speed to create complex queries.

### **A.1    First Normal Form**

The First Normal Form (1NF) ensures that none of the domains of that relation should have elements which are themselves sets. This is encompassed in every relational table so that every entry in the table is unique and there are no duplicate rows.

### **A.2    Second Normal Form**

A table is in a Second Normal Form (2NF) if and only if it is in 1NF and every non-prime attribute of the table is dependent on the whole of a candidate key. In essence, 2NF states that there are no non-trivial functional dependencies on a non-key attribute ensuring that no values are dependent on other values within the row.

### **A.3    Third Normal Form**

Typically, Third Normal Form (3NF) and BCNF prevents transitive dependencies. A transitive dependency is when two columnar relationships imply another relationship. BCNF requires all transitive dependencies be eliminated in addition to the table being 3NF.

### **A.4    Fourth Normal Form**

Fourth Normal Form (4NF) requires that a table be BCNF. 4NF solves the problem of multivalued dependencies, which would be instead encountered when multiple rows were dependent on one another without any clear definition of that dependency in the table. An example would be an encounter with the below pseudo SQL Sales table when a customer purchases multiple products with a single order.



```
CREATE TABLE Sales (  
    customer_name,  
    product_id  
);
```

## **A.5 Fifth Normal Form**

Fifth normal form (5NF), also known as join-projection normal form (JPNF), states that no non-trivial join dependencies exist. 5NF states that any fact should be able to be reconstructed without any anomalous results in any case, regardless of the number of tables being joined. A 5NF table should have only candidate keys and its primary key should consist of only a single column, minimizing the cardinality of the primary key.

## **A.6 Sixth Normal Form**

Sixth Normal Form (6NF), defined by Christopher J. Date in his works (see Reference [7]), is intended to decompose relation variables to irreducible components. This is usually relatively unimportant for non-temporal relation variables, but it can be important when dealing with temporal variables, historical data, or other interval data. For instance, if a relation comprises a supplier's name, status, and city, temporal data may also be added, such as the time during which these values are valid for (eg. for historical data) but the three values may vary independently of each other and at different rates.

## Annex B Star and Snowflake Schema

### B.1 Fact table

According to Reference [8], a star schema as seen in [9] consists of fact tables and dimension tables as exemplified in Figure B.1. Fact tables contain the data relating to the multiple dimensions or factual data of the information being queried. A fact table typically has two types of columns: those that contain facts and those that are foreign keys to dimension tables. The primary key of a fact table is usually a composite key that is made up of all of its foreign keys. This information is often numerical and can consist of several columns and a large magnitude of rows (thousands, millions, billions, etc.).

### B.2 Dimension table

On the other hand, dimension tables are usually smaller and hold descriptive or definition data that reflects the dimensions. SQL queries then use JOIN operations between fact and dimension tables and constraints on the data to return selected information. By using a star schema, databases can be well organized and highly normalized per the operator's requirements.

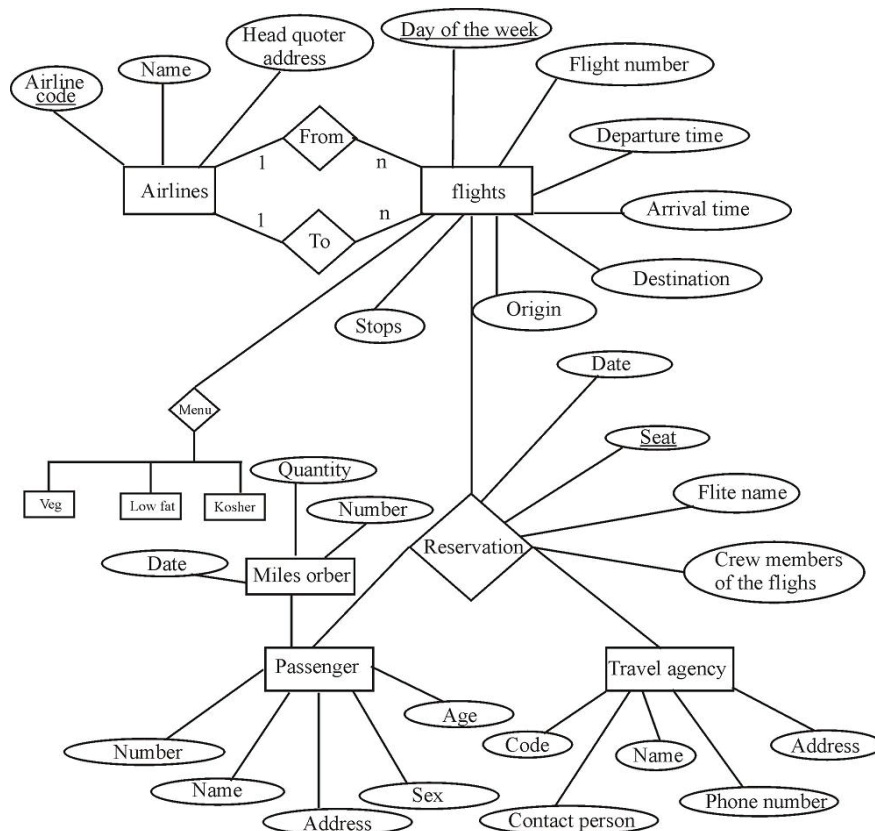


Figure B.1: Example of a Snowflake Schema [9]

## Annex C Anchor Modeling

Described by the project developers at Reference [10], Anchor Modeling is an open source agile database modeling technique suited for information that changes over time both in terms of structure and content. As seen in Figure C.1, it provides the foundation for a graphical notation used for conceptual modeling similar to that of entity-relationship modeling, extendable to working with temporal data. It is built on the premise that the environment surrounding a data warehouse is in constant change. The technique incorporates the natural concepts of objects, attributes and relationships to design a model. This design method results in a highly decomposed implementation, which avoids many of the disadvantages associated with doing the design and normalization manually.

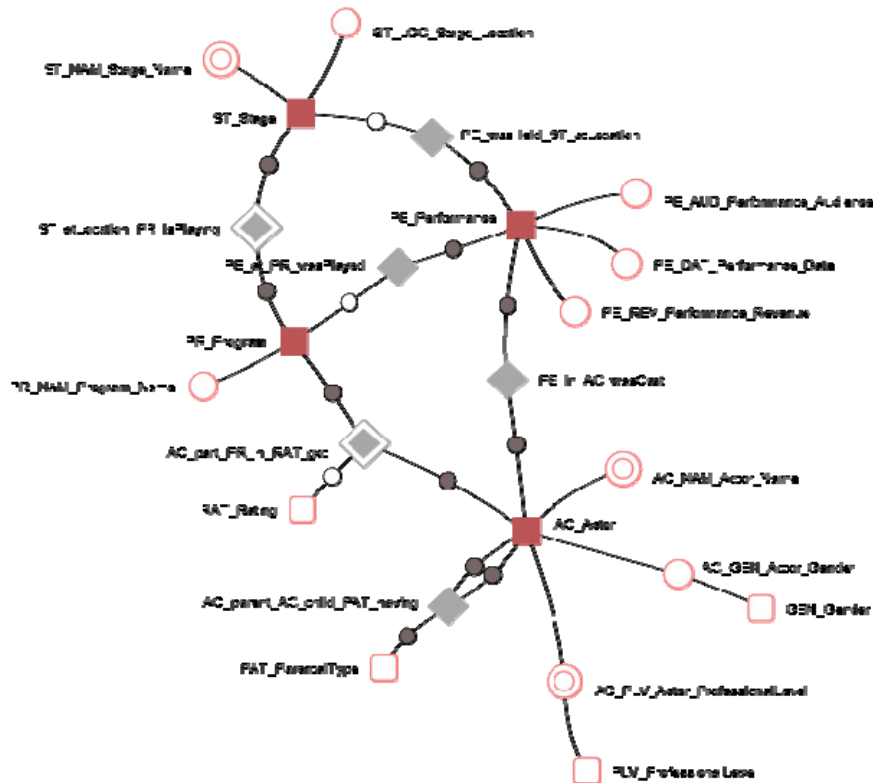


Figure C.1: Anchor Model (See Reference [11])

The modeling technique uses four fundamental modeling entities: anchor, attribute, tie, and knot. Each entity allows the database designer to add different aspects of the data being modeled. Using this structure will allow created models to be translated to physical database designs in SQL using a predefined conversion tool. When translated, the resulting tables in the database will be normalized to the Sixth Normal Form (6NF).

The earliest installations using Anchor Modeling were made in Sweden with the first dating back to 2004, when a data warehouse was built using the technique. In 2007, this database design method was initially presented internationally by Lars Rönnbäck at the TDWI (The Data

Warehousing Institute) conference in Amsterdam. Since then, the research and formalization of Anchor Modeling is being done in collaboration between the creators Olle Regardt and Lars Rönnbäck and a team at the Department of Computer and Systems Sciences from Stockholm University.

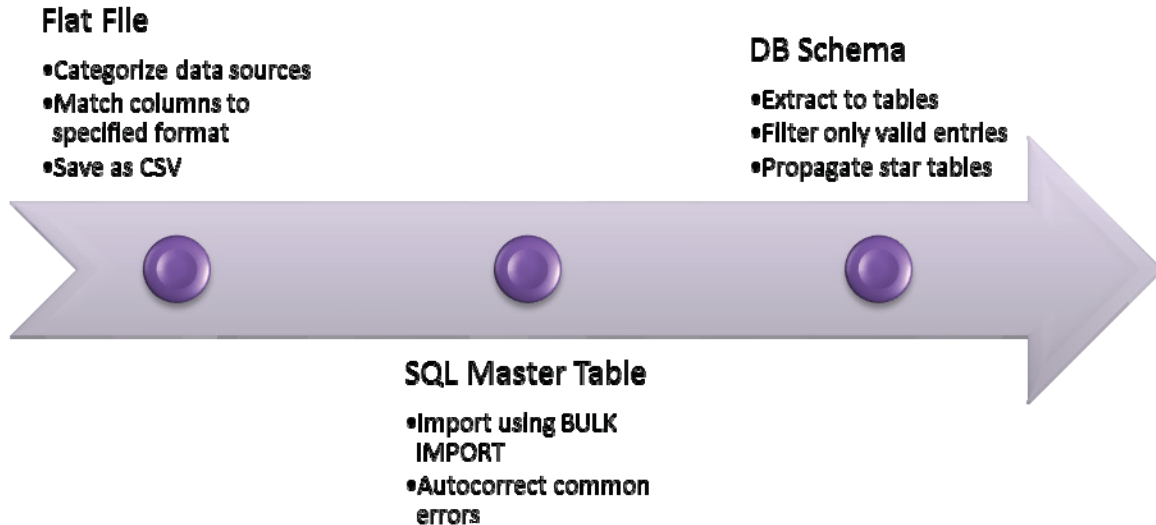
More information can be found at the project's website (<http://www.anchor modeling.com>).



## Annex D Populating the database

---

For the designed database to be useful, data must be populated into the tables. The data import process seen on Figure D.1 illustrates the data formats and operations performed to manipulate the data.



*Figure D.1: Data Import Process*

### D.1 Flat file format

The data values, currently collected by the CAE Professional Services contractors, usually come in an Excel format. The data categories are listed below as follows.

1. Activity
2. Action
3. Position
4. Weather/DMG

The same data categories are separated into different spreadsheets based on different ships, different operations, or a requested date range. In these cases, it is easier to merge spreadsheets of the same data category into one large spreadsheet. However, this is not necessary in all cases as long as the data is formatted into the specified flat file format seen in Figure D.2. It should be

noted that the specified flat file format for each category must be followed; otherwise improper data would be imported into the SQL Master Tables.

Activity	Action	Position	Weather/DMG
<ul style="list-style-type: none"> <li>•Primary Key</li> <li>•Year</li> <li>•Homeport Coast</li> <li>•Ship Abbreviation</li> <li>•Readiness</li> <li>•Activity Remark</li> <li>•Macro Code</li> <li>•Activity Code</li> <li>•Activity Subcode</li> <li>•Duration (Days)</li> <li>•Start Date</li> <li>•Remarks</li> </ul>	<ul style="list-style-type: none"> <li>•Ship Prefix Namesake</li> <li>•Ship Name</li> <li>•Ship Pendant</li> <li>•Ship Commission Date</li> <li>•Ship Class Name</li> <li>•Ship Class Flagship</li> <li>•Ship Commission Status</li> <li>•Action Day</li> <li>•Action Month</li> <li>•Action Year</li> <li>•Origin Port</li> <li>•Origin State</li> <li>•Origin Country</li> <li>•Destination Port</li> <li>•Destination State</li> <li>•Destination Country</li> <li>•Day's Exercise</li> <li>•Action Time</li> <li>•Day's Operation</li> <li>•Manual Classification Code</li> <li>•Remark</li> </ul>	<ul style="list-style-type: none"> <li>•Primary Key</li> <li>•Ship Class Name</li> <li>•Ship Prefix Namesake</li> <li>•Ship Name</li> <li>•Position Date</li> <li>•Ship Status</li> <li>•Position Longitude</li> <li>•Position Latitude</li> <li>•Docked Port</li> <li>•Docked City</li> <li>•Docked State</li> <li>•Docked Country</li> <li>•Docked Sovereign</li> <li>•DataEntryFlag</li> <li>•PositionFlag</li> </ul>	<ul style="list-style-type: none"> <li>•Ship Prefix Namesake</li> <li>•Ship Name</li> <li>•Ship Pendant</li> <li>•Ship Commissioned Date</li> <li>•Ship Class Name</li> <li>•Ship Class Flagship</li> <li>•Weather Day</li> <li>•Weather Month</li> <li>•Weather Year</li> <li>•Ship Port Status (At Sea / Anchor / Moored to Buoy / In Port)</li> <li>•Docked Port</li> <li>•Docked State</li> <li>•Origin Port</li> <li>•Origin State</li> <li>•Destination Port</li> <li>•Destination State</li> <li>•Weather Time</li> <li>•Wind Direction (Degrees)</li> <li>•Wind Speed (Kts)</li> <li>•Sea State</li> <li>•Visibility</li> <li>•Cloud Cover</li> <li>•Barometric Pressure (MilliBars)</li> <li>•Barometric Pressure Direction (Rising/Falling/Steady)</li> <li>•Time Zone</li> <li>•Sunrise Time</li> <li>•Sunset Time</li> <li>•DMG on Middle Watch</li> <li>•DMG on Morning Watch</li> <li>•DMG on Forenoon Watch</li> <li>•DMG on Afternoon Watch</li> <li>•DMG on First Dog Watch</li> <li>•DMG on Last Dog Watch</li> <li>•DMG on First Watch</li> <li>•Remarks</li> </ul>

*Figure D.2: Flat File Format Specification for Data Categories*

The current flat files are separated by the time of which they were obtained by the operator to reduce workload, conflicts, and allow the flat file to be easily created again. The flat file nomenclature is designed to be in alphabetical order and has codename designations as follows:

1. **ForeignKey:** Dataset obtained from the original CAE database project received early May 5, 2013;

2. **MasterKey:** CAE Standing Offer Task 134 data detailing AOR actions, weather, and DMG data on Operation Altair received late May 29, 2013;
3. **PrimaryKey:** CAE Standing Offer Task 146 snapshot detailing FFH actions and DMG data received June 5 2013;
4. **SecondaryKey:** undefined as of writing;
5. **TertiaryKey:** undefined as of writing;

Once the data is formatted correctly and given a unique flat file name, it can be saved into the CSV format for SQL to perform a BULK IMPORT.

## D.2 SQL Master table

SQL Server's BULK IMPORT operator takes a CSV file and imports it into a table. These "Master Tables" have the same columns as the data category specifications in Section D.1. A Master Table's columns are all varchar types to maximize compatibility of incoming data before filtering and further processing. Further instructions on how to use the BULK IMPORT keyword can be found in Reference [12].

An example of SQL code that performs a BULK IMPORT from a CSV flat file can be found below.

```
DELETE
FROM ForeignActions

BULK
INSERT ForeignActions
FROM 'd:\Christopher Woo\NSD Foreignkey - Abbrev.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)

GO
```

Once all data is imported into the MasterTables, SQL can utilize powerful queries to perform correction on the data. Application of data correction methods is highly specific to how the data was corrupted in the first place. For instance, if the column's data type is a time in the DB schema which appears in the format '00:00:00', one instance of corruption is that the data appears as '00;00' (sic). Note that a semicolon was entered instead of a colon preventing the varchar to be casted into a time type. Another example is that valid sea states integers range from 1 to 8. An example of a corrupted sea state value is 13 which is assumed to be a mistyped 3. Finally, data mappings can also constitute as corrupted data because the data does not fit the defined schema. Whereas remarks can be understood by humans, there is no easy way for the computer to easily query a text such as "Low visibility in heavy fog". As such, data columns like visibility must be converted into valid integer values. Again, this decision for conversion depends on the type of



corruption and data type when the data correction is applied. Data correction can be applied when the data is in the Master table or when filtering the data to be extracted into the database schema. Usually more general data correction methods (eg. fixing a time typo) are applied to Master Tables whereas more specific correction methods (eg. mappings) are applied when the data is populated into the database schema.

### D.3 Database schema

With a defined database schema, data can be populated using an INSERT operation which takes values or derived tables and inserts them into a table. An example of the INSERT operation into a table can be seen below. Note that a DELETE operation is performed first to ensure that the previous data is wiped and the table is updated with the latest derived tables from the Master Tables.

```
DELETE FROM ShipAction

INSERT INTO ShipAction
SELECT DISTINCT
    Ship.Id AS ShipId,
    Cast((Cast([Month] As Char(2)) + '/' + Cast([DAY] As Char(2)) +
    '/' + Cast([YEAR] As Char(4)) ) As Date) AS [Date],
    CAST([Time] as Time ) AS [Time],
    Remark
FROM ForeignActions
JOIN Ship
ON      ForeignActions.Name = Ship.Name
JOIN Class
ON      ForeignActions.Class = Class.Name

        AND      Ship.ClassId = Class.Id
```

### D.4 Batch automation

All important INSERT queries, SELECT queries, and analysis queries are saved as SQL script files. The SQLCMD tool that comes with SQL Server Management Studio allows batch files to execute SQL scripts stored in files, automating the import process. Below is the code in Batch that executes 4 SQL scripts: 1 BULK IMPORT scripts for the Master Table, 3 INSERT scripts for derived tables.

```
@echo off
REM Batch import using sqlcmd
echo BULK IMPORT from CSV
sqlcmd -S COR-QGEMINI -i ".\SQL Import\INSERT INTO ForeignActions.sql"
echo.
echo.
echo INSERT FROM raw table
sqlcmd -S COR-QGEMINI -i ".\SQL Import\INSERT INTO ShipActions.sql"
".\SQL Import\INSERT INTO ActionStations.sql" ".\SQL Import\INSERT INTO
ActionVehicle.sql"
```

```
echo.  
echo.  
echo Batch Insert completed  
echo.  
echo.  
PAUSE
```

## **D.5 Digitization**

If a method is developed to digitize content directly from scanned ship logs, then the Flat File and SQL Master Table steps become obsolete. The data can be post processed to ensure correctness and inserted directly into the database tables. This method will improve performance and automate the data import process.



This page intentionally left blank.

DOCUMENT CONTROL DATA		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)  Defence R&D Canada – CORA 101 Colonel By Drive Ottawa, Ontario K1A 0K2	2a. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)  UNCLASSIFIED	
	2b. CONTROLLED GOODS  (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC April 2011	
3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)  Naval Ship Database: Database Design, Implementation, and Schema		
4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)  Woo C. C. Y.		
5. DATE OF PUBLICATION (Month and year of publication of document.)  September 2013	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)  64	6b. NO. OF REFS (Total cited in document.)  12
7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)  Technical Note		
8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)  Defence R&D Canada – CORA 101 Colonel By Drive Ottawa, Ontario K1A 0K2		
9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)	
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)  DRDC CORA TN 2013-157	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)	
11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)  Unlimited		
12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.)  Unlimited		

13. **ABSTRACT** (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This Technical Note documents a database that was designed and implemented to manage a collection of historical data housed by the Maritime Operational Research Team (MORT), which is part of Defence Research and Development Canada's Center for Operational Research and Analysis (DRDC CORA). The goals of the project included: improving integrity; allowing scalability; simplifying queries across existing datasets; and providing schema flexibility for new incoming data. The solution allows database users to store and analyze data collected by navy ships in the Royal Canadian Navy (RCN). The data stored in the database includes but is not limited to ship identification, ship log records, daily scheduled activities, distance made good, designated homeport, geolocation, readiness status, timezone reference changes, and observed weather. Multiple database storage solutions were compared and SQL Server was selected to be the one most suitable for the Naval Ship Database project. Future projects should consider SQL Server as the primary data storage system.

La présente note technique décrit une base de données été conçue et mise en œuvre pour gérer un ensemble de données historiques hébergées par l'équipe de recherche opérationnelle des Forces maritimes, qui fait partie du Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC). Le projet vise à améliorer l'intégrité et l'extensibilité, à simplifier les requêtes dans des ensembles de données existants et à fournir des schémas adaptables qui peuvent accueillir de nouvelles données. La solution mise en place permet aux utilisateurs de stocker et d'analyser des données recueillies par des navires de la Marine royale canadienne (MRC). Sont stockées dans la base de données : données d'identification des navires, entrées de journal de bord, activités quotidiennes prévues, distances couvertes, ports d'attache désignés, géolocalisation, statuts opérationnels, changements de fuseau horaire de référence, données météorologiques observées, etc. Nous avons comparé de multiples solutions de bases de données et déterminé que SQL Server était la solution la mieux adaptée au projet de base de données de navires de la Marine. Les projets à venir devraient envisager l'utilisation de SQL Server comme système principal de stockage de données.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Ship; Database; SQL; Server; Dataset



## **Defence R&D Canada**

Canada's Leader in Defence  
and National Security  
Science and Technology

## **R & D pour la défense Canada**

Chef de file au Canada en matière  
de science et de technologie pour  
la défense et la sécurité nationale



[www.drdc-rddc.gc.ca](http://www.drdc-rddc.gc.ca)

